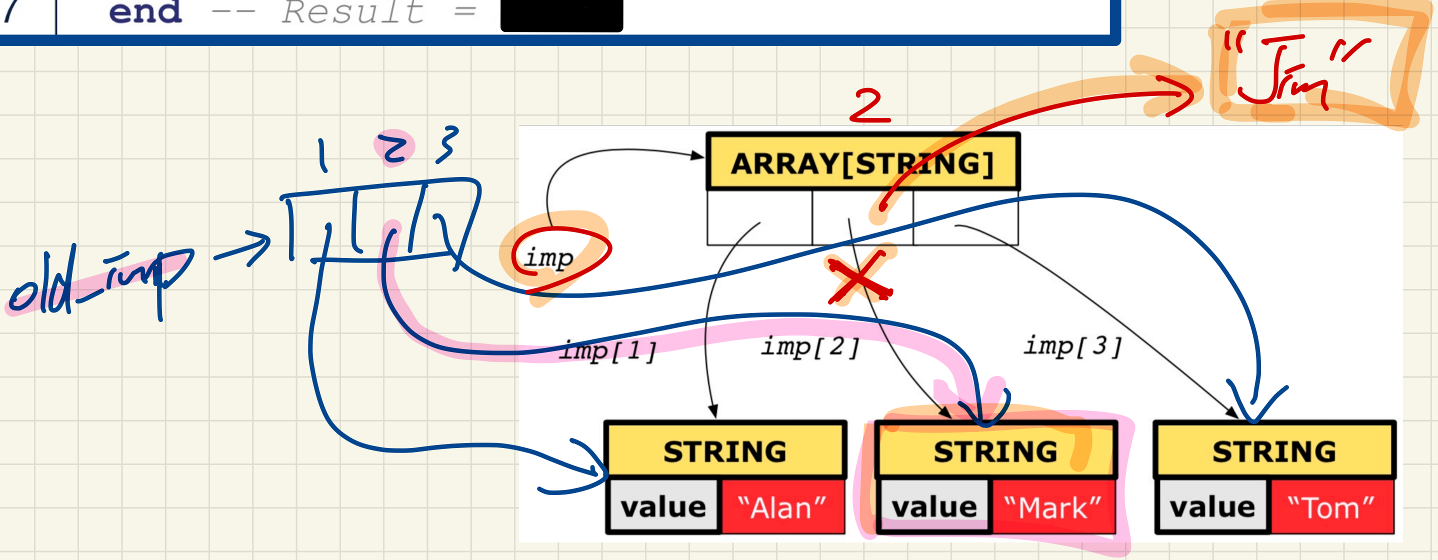
EECS3311 Software Design (Fall 2020)

Q&A - Lecture Series W3

Monday, September 28

Collection Objects: Shallow Copy & Make 1st-Level Changes

```
old_imp := imp.twin
Result := old_imp = imp   -- Result = [redacted]
imp[2] := "Jim"
Result :=
  across 1 |..| imp.count is j
  all imp [j] ~ old_imp [j]
  end -- Result = [redacted]
```



```
old_imp := imp.twin
Result := old_imp = imp   -- Result = [illegible]
~~imp[2] := "Jim"~~   imp.force("Jim", imp.count + 1)
Result :=
  across 1 |..| imp.count is j
  all imp [j] ~ old_imp [j]
  end -- Result = [illegible]
```

3

old_imp.count

→ True but not good

4

4th iteration:

→ imp[4] ~ old_imp[4] → IndexOutOfBound violation.

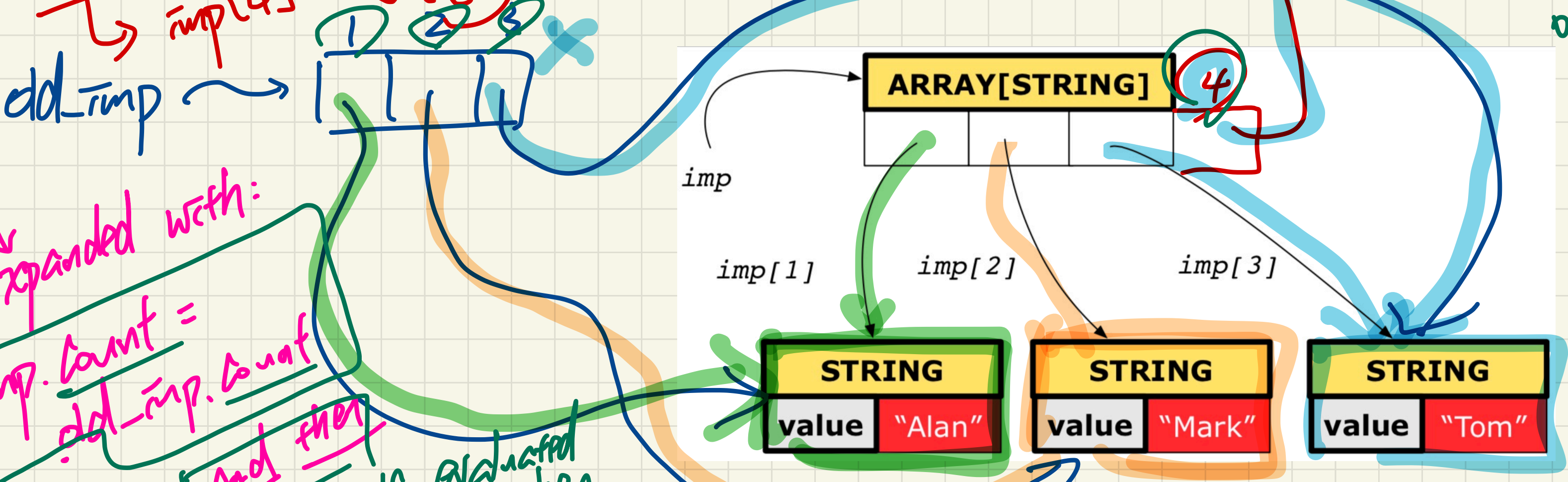
expanded with:

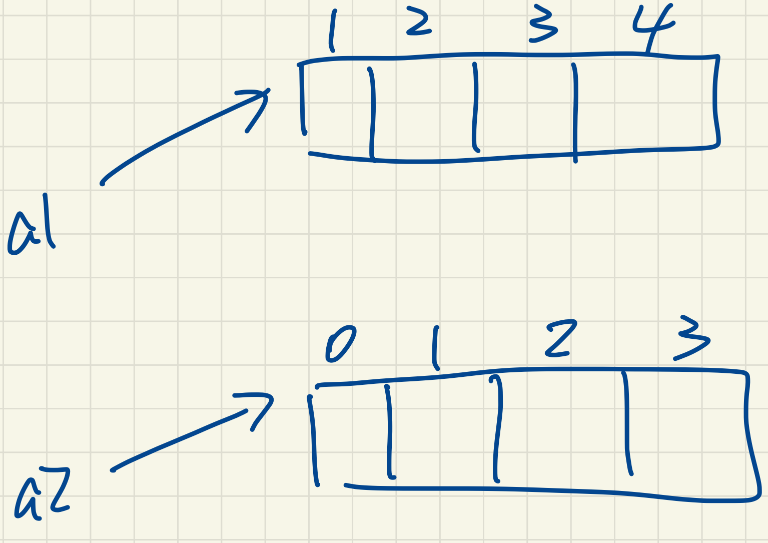
imp.count = old_imp.count

and then

↓ RHS evaluated only when LHS is T

"Jim" ∵ imp[4] is left out.





$a1 \sim a2$ $\boxed{F}$.

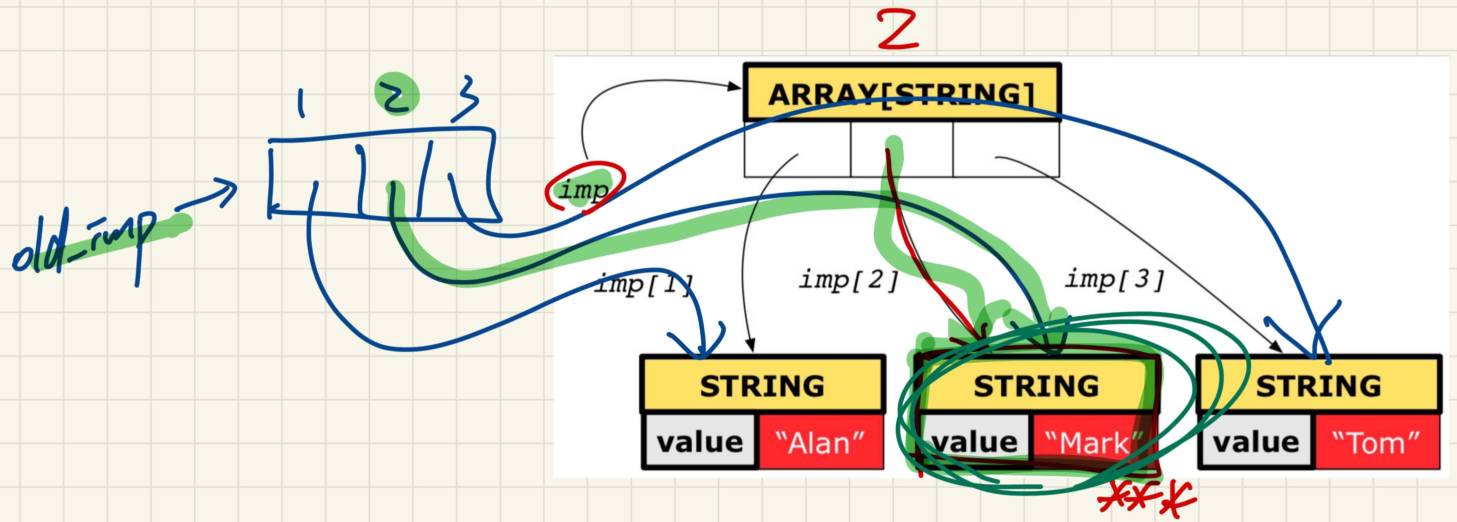
a1.lower = a2.lower
and then
a1.count = a2.count ~~across~~.
and then ?

Collection Objects: Shallow Copy & Make 2nd-Level Changes

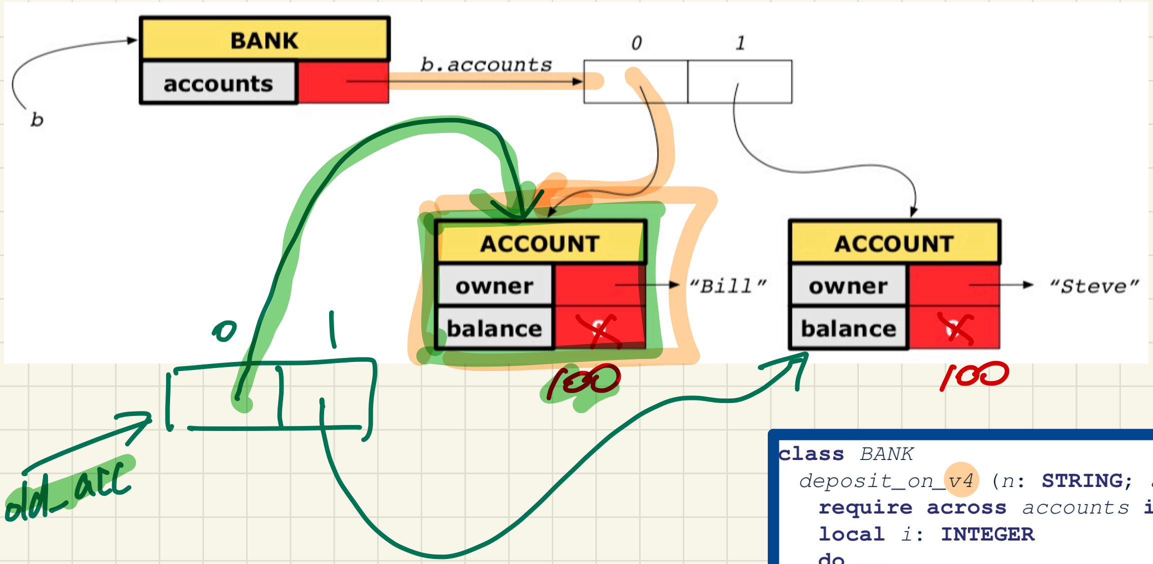
```
old_imp := imp.twin
Result := old_imp = imp   -- Result = [redacted]
imp[2].append ("***")
Result :=
  across 1 |..| imp.count is j
  all imp [j] ~ old_imp [j]
  end -- Result = [redacted]
```

old_imp[2] = imp[2]

T



Version 4: Complete Contracts (Shallow Copy), Wrong Implementation



b.deposit("Steve", 100)

old_acc[0] = b.accounts[0]   T

```
class BANK
  deposit_on_v4 (n: STRING; a: INTEGER)
    require across accounts is acc some acc.owner ~ n end
    local i: INTEGER
    do ...
      -- imp. of version 1, followed by a deposit into 1st account
      accounts[accounts.lower].deposit(a)
    ensure
      num_of_accounts_unchanged: accounts.count = old accounts.count
      balance_of_n_increased:
        Current.account_of(n).balance =
          old Current.account_of(n).balance + a
      others_unchanged :
        across old accounts.twin is acc
        all
          acc.owner /~ n implies acc ~ Current.account_of(acc.owner)
        end
    end
end
```

1st Iteration

acc.owner /~ n **implies** acc ~ **Current**.account_of (acc.owner)

2nd Iteration

acc.owner /~ n **implies** acc ~ **Current**.account_of (acc.owner)

Lab 2.

1. abstraction function model

$$\begin{tikzcd} (\underline{\underline{W3}} \arrow[d] & , W4) \arrow[d] & & \\ SEQ & \boxed{FUN} & \boxed{\underline{REL}} & \\ & & \boxed{W5} & \underline{\underline{L.5}} \end{tikzcd}$$

2. Writing contracts
   ↳ (W4)
      ↳ - assert set equality
        - math. operations

3. <u>Iterator Pattern.</u>

ARRAY.

insert to end or middle

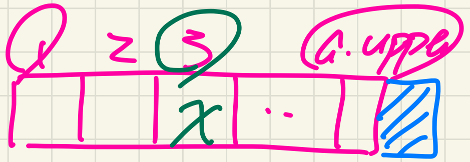
force

a.force(x, 3)

force (v, i)

↳

i ↳ 1 ~ a.upper ↳ put.



a.upper+1 ↳ not a valid index

put

put (v, i)

[ ]

require

a.is_valid_index(i)

↳ a.lower ≤ i ≤ a.upper.

a.put(x, 3)

|||

a[3] := x

```
class STACK[G]
  feature { NONE }  → private
                    → export status
    imp : ~~ARRAY~~ [G]  → Strategy : top is ~~imp[imp.count]~~
          LIST                          imp.first

  feature -- Commands
    push ( v : G )
      ensure
        new_top : top ~ ~~imp[imp.count]~~
                        imp.first.
```

change:
- array → list
- top → front of list

violation of IH principle
∵ strategy of imp. is exposed → likely to change

```
class  STACK [G]
  feature {NONE}
    imp: ~~ARRAY~~ [G]   -- ~~S1~~  S2
         LL
  feature -- public
    model : SEQ[G]
      do
        convert imp ~~array~~ to SEQ
                      LIST
        ensure
          across 1 |..| imp.count is i
          all imp[i] ~ Result[i] end
      end
  feature -- Commands
    push (v: G)
      ensure
        model ~ (old model.deep_twin).appended(v)
```

change:
- array → list
- top → front of list

```
class MY_TABLE [~~B~~ S, ~~A~~ I]
              I    B
create make
feature
    make (x: ~~A~~ I ; y: ~~B~~ S)
             B          I
    do
       ...
    end
```

Which line(s) will compile?

client:

```
t1: MY_TABLE [STRING, INTEGER]
t2: MY_TABLE [INTEGER, BOOLEAN]
```

✓ ① create t1.make ("alan", 3)

③ create t2.make (3, true)

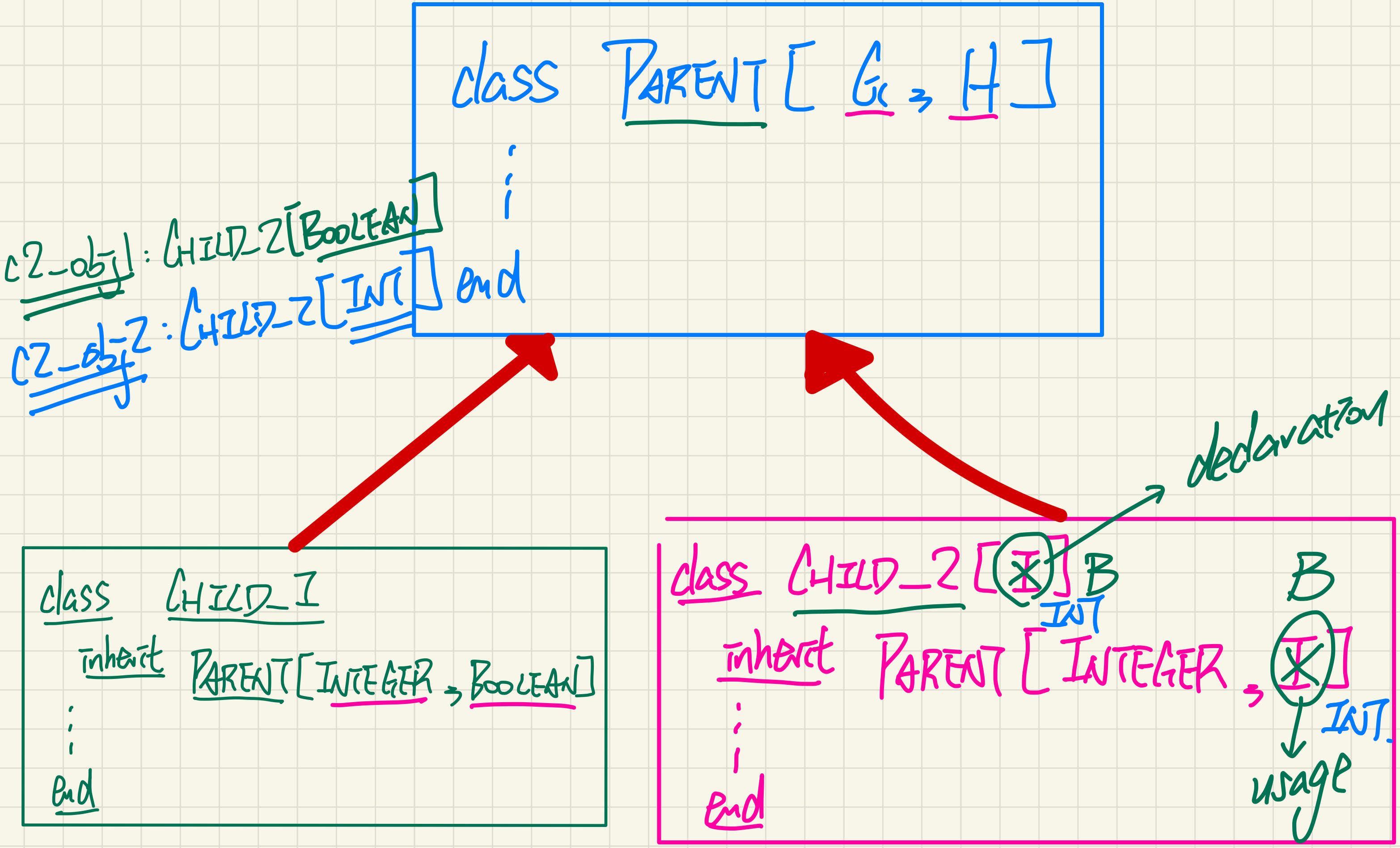
② create t1.make (3, "alan")

④ create t2.make (false, 1)

class MY_CLASS [INTEGER, BOOLEAN] ✗

not compiling

existing classes cannot be used as names of generic parameters.



```
c2_obj1 : CHILD_2[BOOLEAN]
c2_obj2 : CHILD_2[INT]
```

```
class PARENT[G, H]
  :
end
```

```
class CHILD_1
  inherit PARENT[INTEGER, BOOLEAN]
  :
end
```

class CHILD_2[~~X~~] B INT
  inherit PARENT[INTEGER, ~~X~~] B INT
  :
end

cursor

across — as

across — ~~is~~

```
class MY_CLASS [ G -> COMPARABLE ]
```

instantiation must be a descendant class of COMPARA.

Java.
class MyClass<G extends Comparable>

not compile X ~~obj1~~ : MY_CLASS [PERSON]

1 < 2

✓ obj2 : MY_CLASS [INTEGER]

✓ obj3 : MY_CLASS [STRING]

```
class PERSON
inherit
    COMPARABLE
feature
    age : INT.
end
```

is_less_than:

"100" < "alan"

```
class MY_CLASS [ G -> attached ANY ].
```

for concurrent software